The knapsack hash function proposed at crypto'89 can be brocken

Paul Camion, Jacques Patarin

UNITÉ DE RECHERCHE
INRIA-ROCQUENCOURT

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
B.P.105
78153 Le Chesnay Cedex
France
Tél.:(1) 39 63 55 11

Rapports de Recherche

N° 1464

Programme 2
Calcul Symbolique, Programmation
et Génie logiciel

THE KNAPSACK HASH FUNCTION PROPOSED AT CRYPTO'89 CAN BE BROKEN

Paul CAMION
Jacques PATARIN

Juin 1991

RR-1464

La fonction sac à dos de hachage proposée à Crypto'89 peut être cassée

The Knapsack Hash Function proposed at Crypto'89 can be broken

Paul Camion Jacques Patarin

INRIA, B.P. 105, Domaine de Voluceau - Rocquencourt, 78153 Le Chesnay Cedex - France

EUROCRYPT'91, Brighton, England.

Résumé

Ivan Damgård proposa à Crypto'89 des exemples concrets de fonctions de hachage parmi lesquelles un schéma basé sur le sac à dos. Nous allons montrer ici qu'un algorithme probabiliste peut casser ce schéma en un nombre d'opérations élémentaires de l'ordre de 2^{32}. Il est réaliste d'envisager un tel nombre d'opérations sur un calculateur moderne. Donc la fonction de hachage proposée n'est pas très sûre. Un nombre important de ces calculs peut être effectué une fois pour toutes. Un résultat rapide peut être obtenu car une parallélisation est facile à concevoir. De plus des manières de prolonger le présent algorithme à d'autres sac à dos que le (256,128) suggéré par Damgård et analysé ici sont étudiés.

Abstract

Ivan Damgård [4] suggested at Crypto'89 concrete examples of hash functions among which a knapsack scheme. We will here show that a probabilistic algorithm can break this scheme with a number in the region of 2^{32} computations. That number of operations is feasible in realistic time with modern computers. Thus the proposed hash function is not very secure. Among those computations a substantial number can be performed once for all. A faster result can be obtained since parallelism is easy. Moreover, ways to extend the present algorithm to other knapsacks than the present (256, 128) suggested by Damgård are investigated.

The proposed knapsack

Let $a_1, \ldots, a_s$ be fixed integers of A binary digits, randomly selected. If T is a plaintext of s binary symbols, $T = x_1 \ldots x_s$, then $b = \sum_{i=1}^{s} x_i a_i$ will be the proposed hashed value. The values assigned are 256 for s and 120 for A. Thus b has at most $120 + 8 = 128$ binary digits. Thus roughly the probability that a random $256 - bit$ string be a solution is 2^{-128}. We will see that it is somewhat larger. Here a probabilistic algorithm is however designed which solves the problem.
We nevertheless must emphasize that breaking that primitive (the proposed knapsack) does not allow us to break the hash function $h : \{0,1\}^* \rightarrow \{0,1\}^t$ whose construction is described in Theorem 3.1 of [4]. See also [7] for an improvement of that construction. We only show that the family $\mathcal{F}$ of functions defined by the proposed knapsack for $m = 256$ and $t(m) = 128$ is not collision free and hence does not satisfy the hypothesis of Theorem 3.1 of [4].

1 The general scheme of our algorithm

1.1 Description

Aim: Given b, find a binary sequence $x_1, x_2, \ldots, x_s$ such that $\sum_{i=1}^{256} x_i a_i = b$.

This is a Knapsack problem, which has an expected high number of solutions, i.e at first sight $\simeq \frac{2^{256}}{2^{128}}$ and we will show a way to find one of those solutions.

Step 0 : We choose integers m, m_1, m_2, m_3, m_4 such that

a) $m = m_1 m_2 m_3 m_4 > 2^{128}$

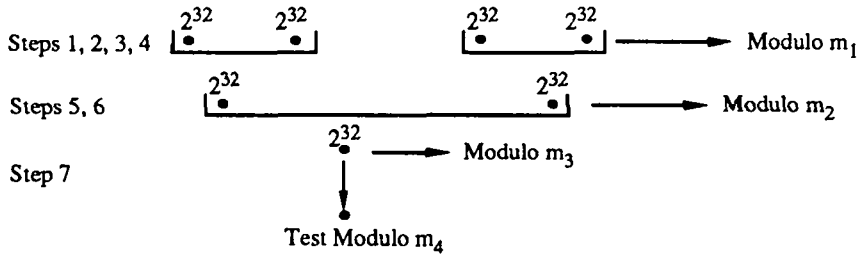
b) The m_i are pairwise coprime, and $m_i \simeq 2^{32}$, $i = 1, 2, 3, 4$.

Let b_i be the residue of b modulo m_i, $i = 1, 2, 3, 4$. By the chinese remainder theorem we have that $x = b$ is the only integer such that

1. $x \equiv b_i \ [m_i], i = 1, 2, 3, 4.$

2. $0 \leq x < m.$

Here is a diagram of the sequence of operations that is considered. Let us sketch the meaning of the diagram before going into details.
Each black point represents a step resulting in an estimated 2^{32} binary sequences. The length of those sequences is 64 for steps 1, 2, 3, 4. It is 128 for steps 5 and 6 and finally step 7 produces about 2^{32} sequences of length 256 among which test modulo m_4 selects a solution with probability close to 1.



We now go through each step in detail.

Step 1 : We find all sequences (x_i), $1 \leq i \leq 64$, $x_i = 0$ or 1, such that

$$\sum_{i=1}^{64} x_i a_i \equiv b_1 \ [m_1].$$

We will find about 2^{32} such sequences because there are 2^{64} sequences (x_i) of 64 bits, and m_1 is close to 2^{32}. In fact, we will examine in Section 2 what is the expected number of solutions to be obtained when the algorithm is brought to completion.

Important remark. Finding sequences (x_i) such that $\sum_{i=1}^{64} x_i a_i \equiv b_1 \ [m_1]$ needs a number in the region of 2^{32} operations, if a memory with sufficient size is available.

Indeed, we just have to do the following.

a) compute and store all values of $b_1 - \sum_{i=1}^{32} x_i a_i$ modulo m_1.

b) compute and store all values of $\sum_{i=33}^{64} x_i a_i$ modulo m_1.

c) keep all pairs of sequences (x_i) which give the same value modulo m_1 in a) and b).

More details. In Section 3 we give more details about operations a), b), c), about the memory needed and about the number of solutions to be found.

Step 2 : The same way, we find about 2^{32} binary sequences (x_i) such that $\sum_{i=65}^{128} x_i a_i \equiv 0 \ [m_1]$.

Step 3 : The same way, we find about 2^{32} binary sequences (x_i) such that $\sum_{i=129}^{192} x_i a_i \equiv 0 \ [m_1]$.

Step 4 : The same way, we find about 2^{32} binary sequences (x_i) such that $\sum_{i=193}^{256} x_i a_i \equiv 0 \ [m_1]$.

Step 5 : We denote $\sum_{i=1}^{64} x_i a_i$ by s_1 and $\sum_{i=65}^{128} x_i a_i$ by s_2.

From the sequences (x_i) found at Steps 1 and 2, and using the procedure a) b) c) above, we find about 2^{32} sequences (x_i), $1 \leq i \leq 128$ such that $s_1 + s_2 \equiv b_2\ [m_2]$. For there are about $2^{32} \times 2^{32} = 2^{64}$ sequences (x_i), $1 \leq i \leq 128$ such that $(x_1, \ldots, x_{64})$ is a solution in Step 1 and $(x_{65}, \ldots, x_{128})$ is a solution in Step 2. So if the numbers $s_1 + s_2$ are about equally distributed modulo $m_2, m_2 \simeq 2^{32}$, we find about $\frac{2^{64}}{2^{32}} = 2^{32}$ among those sequences such that $s_1 + s_2 \equiv b_2\ [m_2]$. If we find noticeably less than 2^{32} such solutions, we will see at the end of this Section 1 what to do.
All sequences (x_i) to be found in Step 5 also have the following property:

$$s_1 + s_2 \equiv b_2\ [m_2] \text{ and } s_1 + s_2 \equiv b_1\ [m_1].$$

This is because $s_1 \equiv b_1\ [m_1]$ and $s_2 \equiv 0\ [m_1]$.

Remark Step 5 also uses a number in the region of 2^{32} operations. The computer problem is the same as for step 1. Details will be given in Section 3. We essentially have two sets E and F of about 2^{32} elements, and we want to find all possible couples (a, b), $a \in E$, $b \in F$, such that $a = b \equiv k\ [m_i]$, where k is a fixed value and m_i is close to 2^{32}.

Step 6 : The same way, from the sequences (x_i) found at Steps 3 and 4, we find about 2^{32} sequences (x_i), $129 \leq i \leq 256$, such that

if $s_3 = \sum_{i=129}^{192} x_i a_i$ and $s_4 = \sum_{i=193}^{256} x_i a_i$, we have that $s_3 + s_4 \equiv 0\ [m_2]$.
Moreover $s_3 + s_4 \equiv 0\ [m_1]$, since $s_3 \equiv 0\ [m_1]$ and $s_4 \equiv 0\ [m_1]$.

Step 7 : The same way, but now from the sequences (x_i) found at the Steps 5 and 6, we find about 2^{32} sequences (x_i), $1 \leq i \leq 256$ such that $(s_1 + s_2) + (s_3 + s_4) \equiv b_3\ [m_3]$.

By construction, we also have that

$$s_1 + s_2 + s_3 + s_4 \equiv b_1\ [m_1] \text{ and } s_1 + s_2 + s_3 + s_4 \equiv b_2\ [m_2].$$

Modulo m_4, we have about 2^{32} possible values, indeed $m_4 \simeq 2^{32}$. Since we found 2^{32} sequences (x_i), there is a high probability that at least one of these sequences is such that $s_1 + s_2 + s_3 + s_4 \equiv b_4\ [m_4]$. We will see in section 2 that we generally find more than one such sequence. We will see at the end of this section what to do if we don't find any such sequence.

Now, suppose that we have such a sequence.

$$s_1 + s_2 + s_3 + s_4 = \sum_{j=1}^{256} x_j a_j.$$

That sequence (x_i) is such that

1. $\sum_{j=1}^{256} x_j a_j \equiv b_i \ [m_i], i = 1, 2, 3, 4.$

2. $0 \leq \sum_{j=1}^{256} x_j a_j \leq 256 \cdot 2^{120} = 2^{128}.$

Thus by a) and b) in Step 0, we finally found a sequence (x_i) such that $\sum_{j=1}^{256} x_j a_j = b$, as desired.
A careful justification of the fact that an average number of 2^{32} solutions are to be found in step 7 is a consequence of corollary 1 in section 2.3. It essentially relies on the fact that the sum of two mutually independent uniformly distributed random variables (m.i.u.d.r.v.) over an abelian group is itself a m.i.u.d.r.v..

Remark What are we going to do if at any a step we have got much less than 2^{32} solutions, or if at the end we don't find any solution such that $s_1+s_2+s_3+s_4 \equiv b_4 \ [m_i], i = 1, 2, 3, 4$?

Then it is possible to use the algorithm again, but with new chosen values. For example we can replace b_1 by $s_1 \equiv b_1 - \lambda \ [m_1]$ at Step 1, and 0 by $s_2 \equiv \lambda \ [m_1]$ at Step 2, where λ is any fixed integer in $[0, m_1 - 1]$.

Or we can even just permute the a_i's.

At each try, we will have a high probability of success, so the probability of success after a few tries can be as close to 1 as desired.

2 Proof

2.1 The collision free functions family

In this section the given sequence element from $[0, 2^{120}[^n$, is denoted by $a^* = \{a_1^*, a_2^*, \ldots, a_n^*\}$ and the value for which a collision is searched is denoted by b^*.
Let us recall what we need form the definition in [5] of a **collision free function family** $\mathcal{F}$. Such a family is given by an infinite family of finite sets $\{F_n\}_{n=1}^{\infty}$, and a function $t : \mathbf{N} \to \mathbf{N}$, such that $t(n) < n$ for all $n \in \mathbf{N}$.

Now a member of F_n here is a function

$$f_a : \{0,1\}^n \to \{0,1\}^t$$

defined by

$$x \rightsquigarrow \sum_{i=1}^{n} x_i a_i$$

where $a = (a_1, \ldots, a_n)$ belongs to the product $\mathcal{A} = [0, 2^{\ell}[^n$ of n intervals of integers, for $\ell = t - \log n$.

Integer $\sum_{i=1}^{n} x_i a_i$ is considered a binary sequence which is its writing in the radix two. Function f_a is called an instance of $\mathcal{F}$ of size n.
We here consider the set F_n for which $n = 256$ and $t(n) = 128$. However the argument here does not generally depend on specifie values of n or other parameters but if so we will point it out.

2.2 Some questions to be replied

The particular function f_a that will be analyzed is given by an a randomly selected from $\mathcal{A}$.
The set $\mathcal{A}$ is thus made a sample space.
For this specific application it has the **equally likely probability measure** but our argument can be exploited in more general situations.
To clarify, let us state one of the problems we will be faced with. Let q be an integer (q is large but small compared to 2^ℓ).
Given b in $[0, 2^t[$, the following expectation will be estimated

$$e(b) = \sum_{a \in \mathcal{A}} p(a) |\{x | x \in \{0,1\}^n, f_a(x) \equiv b \ [q]\}| \tag{1}$$

where p is the probability measure on $\mathcal{A}$.
Other expectations are needed as the following

$$e_j(b) = \sum_{a \in \mathcal{A}} p(a) |\{x | x \in \{0,1\}^n, f_a(x) \equiv b \ [q] \quad \text{and } |s(x)| = j\}|, \ j = 0, \ldots, n,$$

where the **support** $s(x)$ of x is defined as

$$s(x) = \{i | x_i = 1\}.$$

We will in particular observe that those expectations don't depend on a particular choice of a non zero b, for a non zero j.

2.3 Random variables

The boolean function ϕ(statement) takes the value 1 if "statement" is true and 0 otherwise.
We here introduce **random variables** (briefly r.v.) defined on $\mathcal{A}$ and **with values in an abelian group**.
Let q be an integer relatively small compared to 2^ℓ. The additive group $(\mathbb{Z}/q\mathbb{Z}, +)$ of integers modulo q is the abelian group denoted by G_q. Let $I = [0, 2^\ell[$ be a sample space with the equally likely probability measure.
Then the mapping

$$\theta : I \to G_q \tag{2}$$

defines a r.v. which may be considered **uniformly distributed** since q is negligeable compared to $\frac{2^\ell}{q}$.
Moreover we will consider sums of mutually independent (briefly m.i.) r.v. identical to θ and it is not difficult to prove that for whatever probability distribution, (briefly p.d.) at the only condition that the considered random variable with values in an abelian group has a positive probability on each element of a set of generators of that group, then the p.d. of the sum of a large number of m.i.r.v. identical to the given one approaches a uniform distribution.
To every subset J of $[1,n]$ corresponds a random variable $\theta_J : \mathcal{A} \rightarrow G_q$ defined by

$$\forall a \in \mathcal{A} : \theta_J(a) = \sum_{j \in J} a_j \in G_q$$

and we thus have

$$P_r\{\theta_J = b\} = \sum_{a \in \mathcal{A}, \theta_J(a) = b} P(a).$$

For convenience we also define $\theta_\emptyset$ by

$$\forall a \in \mathcal{A} : \theta_\emptyset(a) = 0 \in G_q$$

Clearly θ_J is a sum of $|J|$ m.i.r.v. identical to θ and $\theta_{J_1} = \theta_{J_2}$ for $|J_1| = |J_2|$.
We thus denote by θ_j the common r.v. θ_J with $|J| = j$ and $\theta_\emptyset$ is denoted by θ_0.

We then have that

$$P_r\{\theta_\emptyset = b\} = \phi(b = 0).$$

We now classically have the

Theorem 1 *Given b, then the expectation $e(b)$ is worth*

$$\sum_{a \in \mathcal{A}} p(a) |\{x | x \in \{0,1\}^n, f_a(x) \equiv b \ [q]\}| = \sum_{x \in \{0,1\}^n} P_r\{\theta_{s(x)} = b\}. \tag{3}$$

Proof

The L.H.S. is worth

$$\sum_{(a,x) \in \mathcal{A} \times \{0,1\}^n} p(a) \ \phi\left(\sum_{i=1}^n x_i a_i \equiv b \ [q]\right)$$

$$= \sum_{x \in \{0,1\}^n} \sum_{a \in \mathcal{A}, \sum_{i=1}^n x_i a_i \equiv b \ [q]} p(a),$$

which is nothing but the R.H.S.

Corollary 1 *If θ is a uniformly distributed random variable (briefly u.d.r.v.) then the expectation considered in Theorem 1 is worth*

$$(2^n - 1)/q + \phi(b = 0).$$

Corollary 2 *Given an integer $j, 0 \leq j \leq m$, the expectation*

$$e_j(b) = \sum_{a \in \mathcal{A}} p(a) | \{x | x \in \{0,1\}^n, f_a(x) \equiv b \ [q] \quad and \ |s(x)| = j\}$$

is worth

$$\left(\begin{array}{c} n \\ j \end{array} \right) P_r\{\theta_j = b\}.$$

If θ is a u.d.r.v., then this is $\left(\begin{array}{c} n \\ j \end{array} \right) /q$ *for $j > 0$ and $\phi(b = 0)$ for $j = 0$.*

2.4 The expected weight distribution in the sample to be scanned

2.4.1 The expected number of collisions

Let m_1, m_2, m_3, m_4 be the integers given in section 1 and take $q = m_1 m_2 m_3$.
Since $q \simeq 2^{96} < b^*$ then the set

$$R = \{x \in \{0,1\}^n, f_{a^*}(x) \equiv b^* \ [q]\}$$

contains the set M of all x in $\{0,1\}$ such that $f_{a^*}(x) = b^*$ from which we have to exhibit a member, among other $0-1$ sequences. By Theorem 1 we can assess that the expected size $e(b^*)$ of R is $2^n/q = 2^{160}$.
By Corollary 2, the weight ratio distribution in R is the same as is $\{0,1\}^n$ i.e.

$$\left(\frac{\left(\begin{array}{c} n \\ j \end{array} \right)}{2^n} \right)_{j=0}^{n}$$

The size of M may be in its turn assessed.
The set $\{0,1\}^n$ being assumed to have the equally likely probability measure we have that

$$P_r\{np - t \leq \sum_{i=1}^{n} x_i \leq np + t\} = \sum_{np-t \leq i \leq np+t} \left(\begin{array}{c} n \\ i \end{array} \right) p^i q^{n-i}.$$

For $n = 256, p = q = \frac{1}{2}$ and $t = 20$, this is worth 0.9897. This means that in most cases, b will lie in the interval $[108.2^{119}, 148.2^{119}]$, so that there are about $41.2^{119} \simeq 2^{124.35}$ possible values for b. Thus when a value b^* is assigned to b in that range then the size of M is in

the region of $2^{256-124.35} = 2^{131.64}$.
Now M is contained in R.
This means that the probability to obtain a collision when drawing an element from R at random is $2^{131.64-160} = 2^{-28.35}$.
If we could operate such drawings about 2^{32} times, then the expected number of collisions obtained would be $2^{-28.35} \times 2^{32} = 2^{3.64} > 12$.
Actually our algorithm consists in constructing a subset S of R the expected size of which is 2^{32} in which the expected number of collisions is still $2^{3.64}$.

2.4.2 A suitable partition of the set R

Given a partition $E_1 \cup E_2 \cup E_3 \cup E_4$ of $[1,n]$ with $|E_i| = n/4, i = 1,\ldots,4$ we define a partition $\{S_\gamma\}_{\gamma\in\Gamma}$ of R, where $\Gamma = G_{m_1} \times G_{m_1} \times G_{m_1} \times G_{m_2}, \gamma$ being a quadruple (c_1, c_2, c_3, d_1).
The set S_γ is defined as

$$\{x | x \in R, \sum_{i\in E_j} x_i a_i^* \equiv c_j \ [m_1], j = 1,2,3, \ \sum_{i\in F_1} x_i a_i^* \equiv d_1 \ [m_2]\},$$

where $F_1 = E_1 \cup E_2$. Also $F_2 = E_3 \cup E_4$.

Notice that for x in S_γ, since $S_\gamma \subset R$, we necessarily have that
$\sum_{i\in E_4} x_i a_i^* \equiv b^* - c_1 - c_2 - c_3 \ [m_1]$, $\sum_{i\in F_2} x_i a_i^* \equiv b^* - d_1 \ [m_2]$ and $\sum_{i=1}^{n} x_i a_i^* \equiv b^* [m_3]$.
Clearly by the general property of θ defined in (2.3) the conditional probability

$$P_r\{x \in S_\gamma | x \in R\}$$

is very close to $1/m_1^3 m_2$ and thus Theorem 1 entails that the expected size of S_γ is $|R|/m_1^3 m_2$ which in the example dealt with is 2^{32}.
Notice that in the description of the algorithm, c_1 was chosen to be equal to $b^* mod\ m$, and $c_2 \equiv c_3 \equiv 0 \ [m]$, also $d_1 \equiv b^* \ [m_2]$. **We next give evidence that the expected size of $S_\gamma \cap M$ does not depend on a particular choice of γ.**
We will need a corollary to corollary 2 of Theorem 1.

Corollary 3 *For any $\gamma \in \Gamma$, the weight ratio distribution in S_γ is very close to*

$$\left(\frac{\binom{n}{j}}{2^n} \right)_{j=0}^{n}$$

Indeed for every J for which $J \cap E_i \neq \emptyset$, $i = 1,2,3,4$, we have that

$$P_r(x \in S_\gamma \quad \text{and} \quad s(x) = J)$$
$$= \prod_{j=1}^{4} P_r\left(\sum_{i \in J \cap E_j} x_i a_i \equiv c_j \ [m_1]\right) \prod_{j=1}^{2} P_r\left(\sum_{i \in J \cap F_j} x_i a_i \equiv d_j \ [m_2]\right) P_r\left(\sum_{i=1}^{n} x_i a_i \equiv b \ [m_3]\right)$$

$= 1/m_1^4 m_2^2 m_3$, where $\gamma = (c_1, c_2, c_3, d_1)$ and $c_4 \equiv b - c_1 - c_2 - c_3 \ [m_1]$, $d_2 \equiv b - d \ [m_2]$. Moreover the contribution of the sets J, such that $J \cap E_i = \emptyset$ for some i, to the expectation

$$\sum_{a \in \mathcal{A}} p(a) |\{x | x \in S_\gamma, |s(x)| = j\}|$$

being negligeable for the relevant sizes of j, then a similar argument as the one used for proving corollary 2 proves the thesis.

2.5 Sums of random variables with integer values

The expected size of M, given b, is

$$\sum_{a \in \mathcal{A}} P(a) \left| \left\{ x | x \in \{0,1\}^n, \sum_{i=1}^{n} x_i a_i = b \right\} \right|.$$

A random variable $\omega_J : a \rightsquigarrow \sum_{i \in J} a_i$ mapping $\mathcal{A}$ into $[0, 2^t[$ corresponds to each subset J of $[1,n]$. Notice that $P_r\{\omega_J = b\}$ here depends on b and J when $\mathcal{A}$ has the equally likely probability measure. But by symmetry, for any b, $P_r\{\omega_J = b\}$ only depends on the size of J.

By the same argument as for Theorem 1 we have that the expected size of M is

$$\sum_{J \subset [1,n]} P_r\{\omega_J = b\}.$$

We now consider any partition $H_1 \cup H_2 = [1,n]$. To obtain the evidence claimed in section 2.4.2, we need to show that given q small and any $J \subset [1,n]$ such that $J \cap H_1 \neq \emptyset$, we have that

$$P_r\left\{ \sum_{i \in H_1 \cap J} x_i a_i \equiv c \ [q] \ | x \in M \right\}$$

very slightly depend on the choice of c. Thus the expected size of $M \cap S_\gamma$ will only slightly depend on the choice of γ.

We have that

$$\omega_J(a) = \sum_{i \in J \cap H_1} a_i + \sum_{i \in J \cap H_2} a_i$$

and for $\omega_J(a) = b$, then if $J \cap H_1$ is not empty, $\sum_{i \in J \cap H_1} a_i$ takes all integer values from 0 to b, as a runs over $\mathcal{A}$. We will observe on some numerical values that over q successive values

$v_1, \ldots, v_1 + q - 1$ taken by v in $[0, b]$ then the variation of $P_r\{\omega_{J \cap H_1} = v\}$ is negligeable. Now in view of corollary 3, for a given partition $H_1 \cup H_2 = [1, n]$ where the smallest size of H_i is 64, $i = 1, 2$ then the contribution to

$$P_r\left\{\sum_{i \in H_1} x_i a_i \equiv c\ [q]\ |x \in M\right\}$$

of x's such that $s(x) \cap H_1 = \emptyset$ is negligeable since most of the weights of x's inS_γ are in the interval $[118, 148]$, by corollary 3, and

$$\left(\begin{array}{c} 256 - 64 \\ 118 \end{array}\right) / \left(\begin{array}{c} 256 \\ 118 \end{array}\right) = 8.65\ 10^{-22}.$$

We use the following result on the probability distribution of the sum of j m.i.u.d.r.v. with values in the interval of integers $[0, h[$.
([4] page 52).
Let c be in $[0, jh[$. As c and h grow indefinitely, their ratio $\frac{c}{h}$ approaching ξ which is a real number in $[0, j[$, then the probability distribution

$$a_{j,c} = P_r\{X_1 + X_2 + \ldots + X_j < c\}$$

tends to

$$F(\xi) = \lim_{\frac{c}{h} \to \xi} a_{j,c} = \frac{1}{j!} \sum_{\nu < \xi} \left(\begin{array}{c} j \\ \nu \end{array}\right) (\xi - \nu)^j.$$

We compute $F(\xi)$ and $F(\xi + \varepsilon)$ for $\varepsilon = \frac{q}{h} = \frac{2^{32}}{2^{120}} = 2^{-88}$ and for a few values of j and x. Since $F(\xi)$ is increasing we only have to notice that $F(\xi + \varepsilon)$ is very close to $F(\xi)$ to be convinced of the claim in 2.4.2. We also compute some values for $\varepsilon = 2^{-10}$ to show that for q in the region of 2^{110} we would move away from our assumption.

j	ξ	ε	$F(\xi)$	$F(\xi + \varepsilon) - F(\xi)$	Ratio
32	15	2^{-88}	$2.7\ 10^{-1}$	$6.53\ 10^{-28}$	$2.41\ 10^{-27}$
32	5	2^{-88}	$8.62\ 10^{-14}$	$1.77\ 10^{-39}$	$2.05\ 10^{-26}$
15	7	2^{-88}	$3.28\ 10^{-1}$	$1.04\ 10^{-27}$	$3.15\ 10^{-27}$
15	3	2^{-88}	$1.06\ 10^{-5}$	$1.68\ 10^{-31}$	$1.59\ 10^{-26}$
32	15	2^{-10}	$2.71\ 10^{-1}$	$1.97\ 10^{-4}$	$7.29\ 10^{-4}$
32	5	2^{-10}	$8.62\ 10^{-14}$	$2.17\ 10^{-15}$	$2.51\ 10^{-2}$
15	7	2^{-10}	$3.29\ 10^{-1}$	$3.13\ 10^{-4}$	$9.53\ 10^{-4}$
15	3	2^{-10}	$1.06\ 10^{-5}$	$5.09\ 10^{-8}$	$4.8\ 10^{-3}$

3 More details about the algorithm

3.1 About 2^{32} operations and 64 Gigabytes of memory

At several steps of the algorithm, we are faced with the following problem. We have to find all binary sequences (x_i), $1 \leq i \leq 64$, such that $\sum_{i=1}^{64} x_i a_i \equiv b_1 \ [m_1]$, where $m_1 \simeq 2^{32}$, and where b_1 is a fixed value. We here show in detail how to do this in about 2^{32} operations and we estimate the needed size of memory.

Implementation 1 Let R be a file that can contain 2^{32} words of 32 bits, and let S be a file that can contain 2^{32} words of 64 bits. We store in the file R intermediate results and S will be the file of solutions (x_i).

Step a : For each $(x_1, \ldots, x_{32})$ we compute $k = b_1 - \sum_{i=1}^{32} x_i a_i$ modulo m_1 and store $(x_1, \ldots, x_{32})$ in file R at address k. If a solution $(x'_1, \ldots, x'_{32})$ was already introduced at that address, then $(x_1, \ldots, x_{32})$ is dropped. In implementation 2 however all intermediate results will be stored. This **step a** needs about 2^{32} operations since there are 2^{32} sequences $(x_1, \ldots, x_{32})$.

Step b : For each $(x_{33}, \ldots, x_{64})$ then $k' \equiv \sum_{i=33}^{64} x_i a_i$ modulo m_1 is computed.

First case: The register at address k' in file R contains a sequence $(x_1, \ldots x_{32})$. Then $(x_1, \ldots, x_{32}, x_{33}, \ldots x_{64})$ is stored in S because then $k' \equiv \sum_{i=33}^{64} x_i a_i \ [m_1] \equiv b_1 - \sum_{i=1}^{32} x_i a_i \ [m_1]$, which entails $\sum_{i=1}^{64} x_i a_i \equiv b_1 \ [m_1]$.

Second case: The register at address k' in file R is empty. Here the following $(x_{33}, \ldots, x_{64})$ is considered. This **step b** also needs about 2^{32} operations.

Hence after **a** and **b** we will have a solution (x_i) for each value k' such that $\sum_{i=1}^{64} x_i a_i \equiv b_1 \ [m_1]$ and $\sum_{i=33}^{64} x_i a_i = k'$.

Implementation 2 It is possible to improve implementation 1 in order to obtain all solutions $(x_1, \ldots, x_{64})$ still in about 2^{32} operations and with about $4 \cdot 2^{32} \cdot 32$ bits of memory. Let X be a new file for 2^{32} words of 32 bits.

Step a If the register with address k computed from $(x_1, \ldots, x_{32})$ already contains the intermediate result $(x'_1, \ldots, x'_{32})$, then shift $(x'_1, \ldots, x'_{32})$ to the address $(x_1, \ldots, x_{32})$ of X, and introduce $(x_1, \ldots, x_{32})$ in the register with address k of file R.

Step b We only have to consider the first case. Sequence $(x_1, \ldots, x_{32})$ is found at address k' in file R. Then $(x'_1, \ldots, x'_{32})$ if any is found at address $(x_1, \ldots, x_{32})$ of X. Next $(x''_1, \ldots, x''_{32})$ if any is found at address $(x'_1, \ldots, x'_{32})$ of X and so on untill an empty register appears. In file S are stored successively $(x_1, \ldots, x_{32}, x_{33}, \ldots, x_{64})$,$(x'_1, \ldots, x_{32}, x_{33}, \ldots, x_{64})$, $(x''_1, \ldots, x''_{32}, x_{33}, \ldots, x_{64})$.... That way all solutions are exhibited and stored in S in about 2.2^{32} operations.

Size of memory We need about $4 \cdot 2^{32}.32$ bits = 64 Gigabytes and 1.4 Gigabytes Disks are available today. The size of memory needed is thus high but not unrealistic.

3.2 Some steps can be done once for all

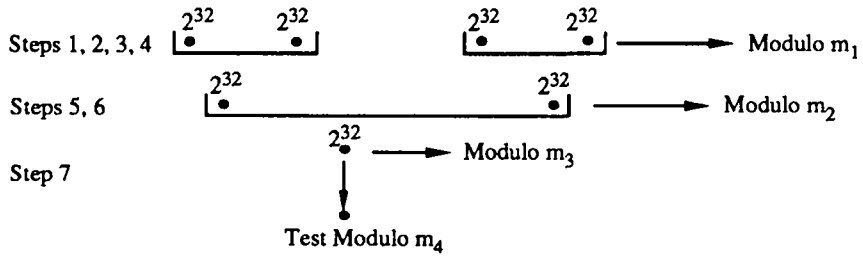
In our algorithm, Steps 2, 3, 4, and 6 do not depend on the value of b. The numbers a_i being publicly disclosed, computations in those steps can be performed once for all.

4 Generalizations of our algorithm to other sizes of Knapsacks

The problem. Let $a_1, \ldots, a_s$, be fixed integers of A binary digits. If T is a plaintext of s binary symbols, i.e. $T = x_1 \ldots x_s$, then $b = \sum_{i=1}^{s} x_i a_i$ is the proposed hashed value. The integer b has B binary digits i.e. $B \simeq A + \log_2 s$.

We have seen in Section 1 that it is possible to find a sequence (x_i) such that $\sum_{i=1}^{s} x_i a_i = b$, when b is given, in about 2^{32} operations, when $s = 256$ and $B = 128$. We will now consider some other values for s and B.

Case $s = 512$ and $B = 160$. It is still possible to find a sequence (x_i) when $s = 512$ and $b = 160$ in about 2^{32} operations: we will just have to add a stage in our algorithm. Schematically we will then have the following. Notice that 2^{32} is the evaluation of the number of solutions.



Explanation of the scheme: m_1, m_2, m_3, m_4, m_5 are pairwise coprime integers such that $m_i \simeq 2^{32}$, $i = 1, 2, 3, 4$, or 5, and $m_1 m_2 m_3 m_4 m_5 > 2^{160}$.

Let b_i be the residue of b modulo m_i.

- At Stage (1), we find about 2^{32} solutions (x_i) such that $\sum_{i=1}^{64} x_i a_i \equiv b_1 \ [m_1]$, about 2^{32} solution (x_i) such that $\sum_{i=65}^{128} x_i a_i \equiv 0 \ [m_1], \ldots$, about 2^{32} solutions (x_i) such that $\sum_{i=449}^{512} x_i a_i \equiv 0 \ [m_1]$.
- At Stage (2), we regroup in pairs the solutions found at Stage (1) to find about 2^{32} solutions (x_i) such that $\sum_{i=1}^{128} x_i a_i \equiv b_2 \ [m_2] \ , \ldots$, about 2^{32} solutions (x_i) such that $\sum_{i=385}^{512} x_i a_i \equiv 0 \ [m_2]$. Notice that these sequences (x_i) also verify $\sum_{i=1}^{128} x_i a_i \equiv b_1 + 0 \ [m_1], \ldots$, and $\sum_{i=385}^{512} x_i a_i \equiv 0 + 0 \ [m_1]$.
- At Stage (3), we regroup in pairs the solutions of Stage (2) to find about 2^{32} solutions (x_i) such that $\sum_{i=1}^{256} x_i a_i \equiv b_3 \ [m_3]$ and 2^{32} solutions (x_i) such that $\sum_{i=257}^{512} x_i a_i \equiv 0 \ [m_3]$. Then at Stage (4) we regroup these solutions to finally find a solution (x_i) at Stage (5) such that $\sum_{i=1}^{512} x_i a_i \equiv b_i \ [m_i]$, $i = 1, 2, 3, 4$ or 5.

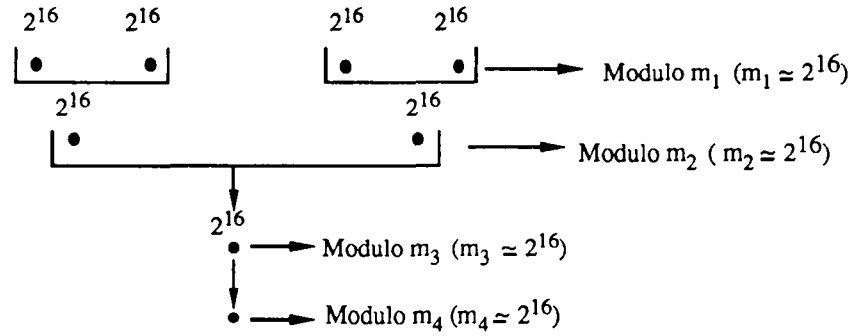
By the chinese remainder theorem, we then found a sequence (x_i) such that $\sum_{i=1}^{512} x_i a_i = b$, as wanted.

Other Generalizations. With the same technique, by using one, two, etc. more stages, and still with about 2^{32} operations, we can solve the cases where $b = 192$ and $s = 1024$, or $b = 224$ and $s = 2048$ etc.

So, our algorithm can solve these cases in about 2^{32} operations:

Upper bound for # binary digits t of b	Lower bound for # binary symbols s of plaintext T.
96	128
128	256
160	512
192	1024
224	2048
etc.	

If $b = 64$ and $s = 128$, using the same process, a sequence (x_i) will be found in about 2^{16} operations. Schematically:



5 Optimality

5.1 A slight improvement

In section 2.4.1, we have seen that the expected number of collisions obtained is at least 12. If the expected size of S_γ defined in 2.4.2 would be 2^{30} instead of 2^{32}, we would have an expected number of at least $2^{16} > 3$ collisions which is a satisfactory prospect.
Let $m_1 = 2^t, m_2 = 2^u, m_3 = 2^v$ and $m^4 = 2^w$. Setting t to 32, u to 31 and v to 32, then by corollary 1, the expected size of S_γ becomes $2^{256-4t-2u-v} = 2^{30}$.

5.2 The designed sizes are best possible. Needlessness of an extra stage

We first point out that if separating step 7 in section 1.1 from the final scanning of S_γ modulo m_4 helps assessing the expectations in section 2, however the result of all computations is the same if we directly compute modulo m_3 m_4 in step 4. Disregarding the refinement above and considering that $2^v = m_3\ m_4$, we have that at the end of the

process the expected number of solutions is 2^r with $r = 256 - 4t - 2u - v$.
The first constraints are thus $t \geq 0, u \geq 0, v \geq 0$ and

$$(1) \qquad 256 - 4t - 2u - v \geq 0$$

But we need that $m_1 m_2 m_3 m_4 \geq b$ for relations 1 and 2 of section 1.1 :

$$(2) \qquad t + u + v \geq 128.$$

Now the expected sizes of the sets we have to deal with are

$$2^{64-t} \quad \text{and} \quad 2^{128-2t-u}.$$

The maximum of these numbers is to be as small as possible and we thus have that

$$(3) \qquad 64 - t = 128 - 2t - u.$$

Adding (1) and (2) we get

$$128 - t - u \geq 2t,$$

and since by (3) we have that $t + u = 64$, we obtain $0 \leq t \leq 32$. We thus minimize 2^{64-t} by setting $t = 32$ and thus $u = 32, v = 64$ which were the selected values. With one more stage, the sizes of the sets to be recorder would be $2^{32-t}, 2^{64-2t-u}, 2^{128-4t-2u-v}$ where $m_1 = 2^t, m_2 = 2^u, m_3 = 2^v, m_4 = 2^w$.
We here have the constraints $t + u + v + w \geq 128$ and $256 - 8t - 4u - 2v - w \geq 0$, from which $128 - 4t - 2u - v \geq 3t + u$. If we attempt to obtain a smaller size than previously, we need that $3t + u \leq 128 - 4t - 2u - v \leq 32$ and $64 - 2t - u \leq 32$, which implies $t \leq 0$. Since $2^t = m_1$ we must have $t = 0$. This means that the considered extra stage is useless.

6 Conclusion

The technique is very efficient for a lot of values of the number s of binary symbols of plaintext T and the number t of binary digits of b. With the values sugested by I. Damgård, we can find a sequence (x_i) using a number in the region of 2^{32} operations and about 60 Gigabytes of memory. This is high, but not impracticable. Still, if $t = 256$ and $s = 512$ for example, finding a sequence (x_i) in a number in the region of 2^{32} operations remains an open problem.

References

[1] P. Camion, *Can a Fast signature Scheme Without Secret Key be Secure?*, in AAECC-2, Lecture Notes in Computer Science n° 228, Springer-Verlag.

[2] P. Camion and Ph. Godlewski, *Manipulation and Errors, Localization and Detection*, Proceedings of EuroCrypt'88, Lecture Notes in Computer Science n° 330, Springer-Verlag.

[3] M. Campana and M. Girault, *How to Use Compressed Encoding Mechanisms in Data Protection*, Securicom 88, March 15–17, pp. 91–110.

[4] D. Dacunha-Castelle and D. Revuz, *Recueil de problèmes de calcul des probabilités*, Masson et Cie, Paris, 1970.

[5] I. Damgård, *Design Principles for Hash Functions*, Proceedings of Crypto'89, Springer-Verlag.

[6] D.W. Davis and W.L. Price, *Security for computer Networks*, John Wiley and Sons, Chichester 1984.

[7] M. Girault, *Hash Functions Using Modulo-n Operations*, Proceedings of EuroCrypt'87, Springer-Verlag.

[8] J.K. Gibson, *Some comments on Damgard's hashing principle*, Electronic letters 19th July 1990, Vol. 26 n° 15.

ISSN 0249-6399